

## APPENDIX 1: SOURCE CODE

**TITLE:** ACCEPTING USER CONTROL  
**APPLICANTS:** NEIL GELFOND, DAMIAN HOWARD, JOSEPH KILLOUGH, ANDREW OLCOTT, PETER C. SANTORO, JAMES SHANLEY AND LEE ZAMIR

Charles Hieken, Reg. No. 18,411  
Fish & Richardson P.C.  
225 Franklin Street  
Boston, MA 02110-2804  
Tel.: (617) 542-5070  
Fax: (617) 542-8906

### CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV 331 654 519 US

April 5, 2004

Date of Deposit

```
*****  
* Name  : Snap  
* Author : Neil Gelfond  
* Notice : Copyright (c) 2002 Bose Corporation  
*          : All Rights Reserved  
* Date   : 08/18/03  
* Version: 1.4  
* Notes  :  
*          :  
*****
```

'This program runs on the PIC Mother Board  
' and send actual ascii codes

```
'cmcon    = 7      'disable comparitors  
OPTION_REG.7 = 0      'enable weak pull-ups  
adcon1    = $0e
```

```
switch  var porta.0
```

```
tx      var portb.2  
tx2     var portb.7  
mode    var word
```

```
'mode = 84      '9600 non-inverted  
'mode = 16468   '9600 inverted  
mode = 188      '4800 non-inverted  
'mode = 16572   '4800 inverted
```

```
define adc_bits 8  
define adc_clock 3  
define adc_sampleus 50
```

```
value    var byte  
res_value  var byte  
zone    var byte  
prev_zone  var byte  
current_zone var byte  
zone_buff  var byte[3]
```

```
input switch
```

```
'define hser_rcsta 90h  
'define hser_txsta 20h  
'define hser_baud 4800  
'define hser_spbrg 12
```

pinA var porta.3  
pinB var porta.4

pinC var portb.0  
pinD var portb.1

input pinA  
input pinB  
input pinC  
input pinD

counterA var word  
counterB var word  
counterC var word  
counterD var word

bufcount var word  
ticcount var byte

tempstateA var bit  
tempstateB var bit  
tempstateC var bit  
tempstateD var bit

prevstateA var bit  
prevstateB var bit  
prevstateC var bit  
prevstateD var bit

stateA var bit  
stateB var bit  
stateC var bit  
stateD var bit

flagA var bit  
flagAP var bit  
flagB var bit  
flagBP var bit  
flagC var bit  
flagCP var bit  
flagD var bit  
flagDP var bit

limit con 3  
PlusMinus var byte

```
old_PlusMinus var byte
buffer      var byte[3]
dir         var byte
pulsedata   var byte
```

```
ticcount = 0
dir      = 0
pulsedata = 0
```

```
flagA    = 0
flagAP   = 0
```

```
flagB    = 0
flagBP   = 0
```

```
flagC    = 0
flagCP   = 0
```

```
flagD    = 0
flagDP   = 0
```

```
stateA   = pinA
stateB   = pinB
stateC   = pinC
stateD   = pinD
```

```
prevstateA = pinA
prevstateB = pinB
prevstateC = pinC
prevstateD = pinD
```

```
tempstateA = pinA
tempstateB = pinB
tempstateC = pinC
tempstateD = pinD
```

```
counterA  = limit
counterB  = limit
counterC  = limit
counterD  = limit
*****ZoneInit*****
```

```
zone_buff(0) = $ff
zone_buff(1) = $ff
zone_buff(2) = $ff
zone      = 0
```

```
prev_zone = 0
current_zone = 0

ADCIN 2,value

value = value
res_value = value

""""bufinit"""
bufinit:
buffer(0) = $ff
buffer(1) = $ff
buffer(2) = $ff
bufcount = 0
""""main"""
main:

'gosub initialize

gosub Rotory

gosub Direction

gosub BufStuff

gosub Display

"""

gosub resvalue

gosub Look

gosub a2dbuffer

gosub Press_Test

gosub Release_test

goto main
""""initialize"""
initialize:

stateA = pinA
stateB = pinB
stateC = pinC
```

```

stateD = pinD
prevstateA = pinA
prevstateB = pinB
prevstateC = pinC
prevstateD = pinD

return
*****"Rotory"*****
Rotory:

gosub CheckPinA

gosub CheckPinB

gosub CheckPinC

gosub CheckPinD

if (flagA = 1) or (flagB = 1) then
  pulsedata.0 = prevstateA
  pulsedata.1 = prevstateB
  pulsedata.2 = stateA
  pulsedata.3 = stateB
  flagA = 0
  flagB = 0
  bufcount = 0
else
  if (flagC = 1) or (flagD = 1) then
    pulsedata.0 = prevstateC
    pulsedata.1 = prevstateD
    pulsedata.2 = stateC
    pulsedata.3 = stateD
    flagC = 0
    flagD = 0
    bufcount = 0
  endif
endif

exit_rotory:

return
*****"CheckPinA"*****
CheckPinA:

if counterA = limit then
  goto statecheckA

```

```

else
  counterA = counterA +1
  goto countercheckA
endif

countercheckA:
if counterA = limit then
  if tempstateA <> pinA then
    goto resetcounterA
  else
    prevstateA = stateA
    stateA = tempstateA
    flagA = 1
    flagAP = 1
    goto exitA
  endif
else
  goto tempstatecheckA
endif

tempstatecheckA:
if tempstateA <> pinA then
  goto resetcounterA
else
  goto exitA
endif

statecheckA:
if stateA <> pinA then
  goto resetcounterA
else
  goto exitA
endif

resetcounterA:
flagA = 0
counterA = 0
tempstateA = pinA

exitA:
return
""""""""CheckPinB"""""""""
CheckPinB:

if counterB = limit then
  goto statecheckB

```

```

else
  counterB = counterB +1
  goto countercheckB
endif

countercheckB:
if counterB = limit then
  if tempstateB <> pinB then
    goto resetcounterB
  else
    prevstateB = stateB
    stateB = tempstateB
    flagB = 1
    flagBP = 1
    goto exitB
  endif
else
  goto tempstatecheckB
endif

tempstatecheckB:
if tempstateB <> pinB then
  goto resetcounterB
else
  goto exitB
endif

statecheckB:
if stateB <> pinB then
  goto resetcounterB
else
  goto exitB
endif

resetcounterB:
flagB = 0
counterB = 0
tempstateB = pinB

exitB:
return
*****CheckPinC*****
CheckPinC:

if counterC = limit then
  goto statecheckC

```

```

else
  counterC = counterC +1
  goto countercheckC
endif

countercheckC:
if counterC = limit then
  if tempstateC <> pinC then
    goto resetcounterC
  else
    prevstateC = stateC
    stateC = tempstateC
    flagC = 1
    flagCP = 1
    goto exitC
  endif
else
  goto tempstatecheckC
endif

tempstatecheckC:
if tempstateC <> pinC then
  goto resetcounterC
else
  goto exitC
endif

statecheckC:
if stateC <> pinC then
  goto resetcounterC
else
  goto exitC
endif

resetcounterC:
flagC = 0
counterC = 0
tempstateC = pinC

exitC:
return
""""""""CheckPinD"""""""""""
CheckPinD:

if counterD = limit then
  goto statecheckD

```

```

else
  counterD = counterD +1
  goto countercheckD
endif

countercheckD:
if counterD = limit then
  if tempstateD <> pinD then
    goto resetcounterD
  else
    prevstateD = stateD
    stateD = tempstateD
    flagD = 1
    flagDP = 1
    goto exitD
  endif
else
  goto tempstatecheckD
endif

tempstatecheckD:
if tempstateD <> pinD then
  goto resetcounterD
else
  goto exitD
endif

statecheckD:
if stateD <> pinD then
  goto resetcounterD
else
  goto exitD
endif

resetcounterD:
flagD = 0
counterD = 0
tempstateD = pinD

exitD:
return
"*****Buffer Stuff*****"
BufStuff:

if (buffer(1) = $ff) or (buffer(2) = $ff) then
  'hserout [hex2 buffer(0)," ",hex2 buffer(1)," ",hex2 buffer(2),13,10]

```

```
    goto shift
  else
    dir.0 = buffer(0)
    dir.1 = buffer(1)
    dir.2 = buffer(2)
  endif

  gosub ClockCounterClock

  shift:
  buffer(2) = buffer(1)
  buffer(1) = buffer(0)
  'hserout [hex2 buffer(0)," ",hex2 buffer(1)," ",hex2 buffer(2),13,10]

  return

"Direction"
Direction:

  select case pulsedata

  case 0,5,10,15
    buffer(0) = $ff

  case 1,7,8,14
    buffer(0) = 1

  case 2,4,11,13
    buffer(0) = 0

  case 3,6,9,12
    buffer(0) = $ff

  end select

  return
"ClockCounterClock"
ClockCounterClock:

  select case dir

  case 0,1,2,4
    PlusMinus = 45 '(- counter-clock-wise)

  case 3,5,6,7
    PlusMinus = 43 '(+ clock-wise)
```

```

end select

return
""""""Display"""""
Display:

if (flagAP = 1) and (flagBP = 1) then
  if old_PlusMinus <> PlusMinus then
    'ticcount = 0
    old_PlusMinus = PlusMinus
  endif
  'ticcount = ticcount + 1

  if PlusMinus = 45 then
    'high left
    ticcount = ticcount - 1
    if ticcount = 0 or ticcount = 255 then
      ticcount = 30
    'ticcount = 16
    endif
    serout2 tx,mode,[\$24,\$4c,\$2b,\$2a]
    'serout2 tx2,mode,[\$24,\$4c,\$2b,\$2a]
    'serout2 tx,mode,[dec ticcount,13,10] '[PlusMinus,13,10]
    'hserout [dec ticcount,13,10] '[PlusMinus,13,10]
    'pauseus 10000
    'low left
  else
    'high right
    ticcount = ticcount + 1
    if ticcount = 31 then
      'if ticcount = 17 then
        ticcount = 1
      endif
      serout2 tx,mode,[\$24,\$4c,\$2d,\$2a]
      'serout2 tx2,mode,[\$24,\$4c,\$2d,\$2a]
      'serout2 tx,mode,[dec ticcount,13,10] '[PlusMinus,13,10]
      'hserout [dec ticcount,13,10] '[PlusMinus,13,10]
      'pauseus 10000
      'low right
    endif
  gosub initialize

  flagAP = 0
  flagBP = 0

```

```
endif
```

```
if (flagCP = 1) and (flagDP = 1) then
  if old_PlusMinus <> PlusMinus then
    'ticcount = 0
    old_PlusMinus = PlusMinus
  endif
  'ticcount = ticcount + 1

  if PlusMinus = 45 then
    'high left
    ticcount = ticcount - 1
    if ticcount = 0 or ticcount = 255 then
      ticcount = 30
      'ticcount = 16
    endif
    serout2 tx,mode,[\$24,\$52,\$2b,\$2a]
    'serout2 tx2,mode,[\$24,\$52,\$2b,\$2a]
    'serout2 tx,mode,[dec ticcount,13,10] '[PlusMinus,13,10]
    'hserout [dec ticcount,13,10] '[PlusMinus,13,10]
    'pauseus 10000
    'low left
  else
    'high right
    ticcount = ticcount + 1
    if ticcount = 31 then
      'if ticcount = 17 then
        ticcount = 1
      endif
      serout2 tx,mode,[\$24,\$52,\$2d,\$2a]
      'serout2 tx2,mode,[\$24,\$52,\$2d,\$2a]
      'serout2 tx,mode,[dec ticcount,13,10] '[PlusMinus,13,10]
      'hserout [dec ticcount,13,10] '[PlusMinus,13,10]
      'pauseus 10000
      'low right
    endif
  gosub initialize

  flagCP = 0
  flagDP = 0
  endif
```

Return  
""""""Resvalue""""""  
Resvalue:

```
ADCIN 0,value
if res_value <> value then
  pause 50
  res_value = value
  goto resvalue
endif
```

```
res_value = value
```

```
exit_resval:
```

```
return
```

""""""Look up"""""""
Look:

```
select case res_value
'serout2 tx,mode,["Res Value ",dec res_value,13,10]

case 104,105,106      'top left ,232
  zone_buff(0) = $31
  'zone = 1

case 92,93,94,95    'left knob
  zone_buff(0) = $36
  'zone = 6

case 178,179,180,181      'bottom left
  zone_buff(0) = $32
  'zone = 2

case 117,118,119,120,121      'Top Center
  zone = $35
  'zone = 5

case 97,98,99,100,101      'Bottom Center
  zone = $38
  'zone = 8
```

```

case 210,211,212,213,214      'top right
  zone_buff(0) = $33
  'zone = 3

case 129,130,131      'Right knob 157,118,162,106,166,165,117
  zone_buff(0) = $37
  'zone = 7

case 141,142,143,144      'bottom right
  zone_buff(0) = $34
  'zone = 4

'case is < 20
 ' zone = 0

case else
  zone_buff(0) = 0
end select

```

exit\_lookup:

return

""""""A2D Buffer""""""""

a2dbuffer:

```

if (zone_buff(1) = $ff) or (zone_buff(2) = $ff) then
  goto buff_shift
endif

```

```

      """"compare buffers"""""""
if zone_buff(0) = zone_buff(1) then
  zone = zone_buff(0)
  goto clear_buff
endif

```

```

if zone_buff(0) = zone_buff(2) then
  zone = zone_buff(0)
  goto clear_buff
endif

```

```

if zone_buff(1) = zone_buff(2) then
  zone = zone_buff(1)
  goto clear_buff
endif

```

```
""""""clear buffers"""""""
clear_buff:
zone_buff(0) = $ff
zone_buff(1) = $ff
zone_buff(2) = $ff
pauseus 100
goto exit_a2dbuffer
```

```
buff_shift:
zone_buff(2) = zone_buff(1)
zone_buff(1) = zone_buff(0)
```

```
exit_a2dbuffer:
```

```
return
""""""Press Test"""""""
Press_Test:
```

```
if zone <> current_zone then
  if zone <> 0 then
    current_zone = zone
    gosub press
  endif
endif
```

```
exit_press_test:
return
""""""Release Test"""""""
Release_Test:
```

```
if zone <> current_zone then
  gosub release
  current_zone = zone
endif
```

```
exit_rel_test:
return
""""""Display Press"""""""
Press:
```

```
serout2 tx,mode,[\$24,zone,\$50,\$2a]
'serout2 tx2,mode,[\$24,zone,\$50,\$2a]
```

```
serout2 tx2,mode,[ "Res Value ",dec res_value,13,10]
'serout2 tx,mode,[ "Z ",dec zone," Pr ",13,10]
```

```
exit_press:
return
```

```
""""""""Display Release"""""""""
Release:
```

```
serout2 tx,mode,[ $24,current_zone,$52,$2a]
'serout2 tx2,mode,[ $24,current_zone,$52,$2a]
'serout2 tx,mode,[ "Z ",dec current_zone," Rel ",13,10]
'serout2 tx,mode,[ " ",13,10]
```

```
exit_release:
return
```